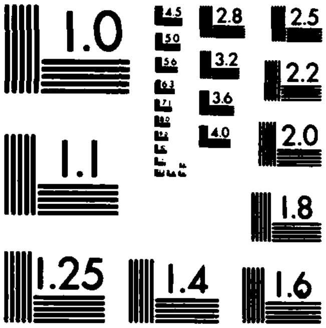
AD-A132 939 USER'S GUIDE FOR SOL/NPSOL: A FORTRAN PACKAGE FOR NONLINEAR PROGRAMMING(U) STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB P E GILL ET AL. JUL 83 SOL-83-12 1/1

UNCLASSIFIED ARO-18424.12-MA N00014-75-C-0267 F/G 12/1 NL

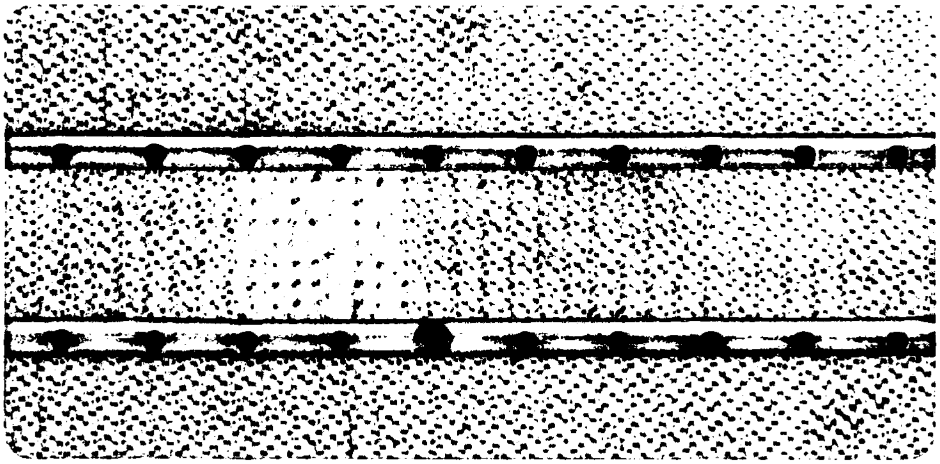


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ARO 18424.12-MA

12

Systems
Optimization
Laboratory



Department of Operations Research
Stanford University
Stanford, CA 94305

12

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

User's Guide for SOL/NPSOL[†]: A Fortran Package for Nonlinear Programming

by

Philip E. Gill, Walter Murray,
Michael A. Saunders and Margaret H. Wright

TECHNICAL REPORT SOL 83-12

July 1983

[†]SOL/NPSOL is available from the Office of Technology Licensing, 105 Encina Hall, Stanford University, Stanford, California, 94305.

Research and reproduction of this report were partially supported by the National Science Foundation Grants MCS-7926009 and ECS-8012974; Department of Energy Contract DE-AM03-76SF00326, PA# DE-AT03-76ER72018; Office of Naval Research Contract N00014-75-C-0267; and Army Research Office Contract DAAG29-81-K-156.

User's Guide for SOL/NPSOL†: a Fortran Package for Nonlinear Programming

Philip E. Gill, Walter Murray,
Michael A. Saunders and Margaret H. Wright
Systems Optimization Laboratory
Department of Operations Research
Stanford University
Stanford, California 94305

July 1983

ABSTRACT

This report forms the user's guide for Version 1.1 of SOL/NPSOL, a set of Fortran subroutines designed to minimize an arbitrary smooth function subject to constraints, which may include simple bounds on the variables, linear constraints and smooth nonlinear constraints. (NPSOL may also be used for unconstrained, bound-constrained and linearly constrained optimization.) The user must provide subroutines that define the objective and constraint functions and their gradients. All matrices are treated as dense, and hence NPSOL is not intended for large sparse problems.

NPSOL uses a sequential quadratic programming (SQP) algorithm, in which the search direction is the solution of a quadratic programming (QP) subproblem. The algorithm treats bounds, linear constraints and nonlinear constraints separately. The Hessian of each QP subproblem is a positive-definite quasi-Newton approximation to the Hessian of an augmented Lagrangian function. The steplength at each iteration is required to produce a sufficient decrease in an augmented Lagrangian merit function. Each QP subproblem is solved using a quadratic programming package with several features that improve the efficiency of an SQP algorithm.

---

†The package SOL/NPSOL is available from the Office of Technology Licensing, 105 Encina Hall, Stanford University, Stanford, California, 94305.

The material contained in this report is based upon research supported by the U.S. Department of Energy Contract DE-AC03-76SF00326, PA No. DE-AT03-76ER72018; National Science Foundation Grants MCS-7926009 and ECS-8012974; the Office of Naval Research Grant N00014-75-C-0267; and the U.S. Army Research Office Contract DAAG29-79-C-0110.

NPSOL User's Guide

TABLE OF CONTENTS

Acknowledgement

The authors wish to thank Klaus Schittkowski for the benefits derived from his visit to the Systems Optimization Laboratory during the early work on NPSOL. We are especially grateful for the set of test problems that he provided, and for discussions of the SQP algorithm in his nonlinear programming code NLPQL. In particular, we have followed his use of an augmented Lagrangian merit function.

1. PURPOSE

SOL/NPSOL is a collection of Fortran subroutines designed to solve the *nonlinear programming problem* — the minimization of a smooth nonlinear function subject to a set of constraints on the variables. The problem is assumed to be stated in the following form:

$$\text{NP} \qquad \underset{x\in\Re^n}{\text{minimize}} \quad F(x)$$
$$\text{subject to} \quad \ell \le \left\{ \begin{matrix} x \\ A_L x \\ c(x) \end{matrix} \right\} \le u,$$

where $F(x)$ is a smooth nonlinear function, A_L is a constant matrix, and $c(x)$ is a vector of smooth nonlinear constraint functions. The matrix A_L and the vector $c(x)$ may be empty. Note that *upper and lower bounds are specified for all the variables and for all the constraints.* This form allows full generality in specifying other types of constraints. In particular, the i-th constraint may be defined as an *equality* by setting $\ell_i = u_i$. If certain bounds are not present, the associated elements of ℓ or u can be set to special values that will be treated as $-\infty$ or $+\infty$.

If no nonlinear constraints are present, it is generally more efficient to use a package specifically designed for linearly constrained problems. In particular, when F is linear or quadratic, the LPSOL or QPSOL packages should be used (Gill *et al.*, 1983a); for a general function F with only linear constraints, the LCSOL package is appropriate (Gill *et al.*, 1983c). If the problem is large and sparse, the MINOS/AUGMENTED package (Murtagh and Saunders, 1980, 1982) should be used, since NPSOL treats all matrices as dense.

The user must supply an initial estimate of the solution to NP, and subroutines that define $F(x)$, $c(x)$ and their first derivatives. The level of printed output is controlled by the user (see the parameter MSGLVL in Section 4).

NPSOL is based on subroutines from Version 3.1 of the SOL/QPSOL quadratic programming package; the documentation of this version of QPSOL (Gill *et al.*, 1983a) should be consulted in conjunction with this report. NPSOL contains approximately 9000 lines of ANSI (1966) Standard Fortran, of which 47% are comments.

2. DESCRIPTION

The method used to solve NP is a sequential quadratic programming (SQP) method. SQP methods were popularized mainly by Biggs (1972), Han (1976) and Powell (1977); for an overview, see, e.g., Fletcher (1981), Gill, Murray and Wright (1981) and Powell (1982). Let x_0 denote the initial estimate of the solution. During the k-th "major iteration" of NPSOL ($k = 0, 1, \ldots$), a new estimate is defined by

$$x_{k+1} = x_k + \alpha_k p_k,$$

where the vector p_k is the solution of a QP subproblem, to be described below. The positive scalar α_k is chosen to produce a sufficient decrease in an augmented Lagrangian merit function (see Schittkowski, 1981); the procedure that determines α_k is called the *line search*.

The QP subproblem that defines p_k is of the form

$$\text{QP} \qquad \underset{p\in\Re^n}{\text{minimize}} \quad g^Tp + \frac{1}{2}p^THp$$
$$\text{subject to} \quad \bar{l} \le \begin{Bmatrix} p \\ \mathcal{A}p \end{Bmatrix} \le \bar{u}.$$

The vector g in QP is the gradient of F at x_k. The matrix H is a positive-definite quasi-Newton approximation to the Hessian of an augmented Lagrangian function. It is represented as $H = R^TR$, where R is upper triangular, and is updated after every major iteration.

Let m_L denote the number of linear constraints (the number of rows in $\mathcal{A}_L$), and let m_N denote the number of nonlinear constraints (the dimension of $c(x)$). The matrix $\mathcal{A}$ in QP has $m_L + m_N$ rows, and is defined as

$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_L \\ \mathcal{A}_N \end{pmatrix},$$

where $\mathcal{A}_N$ is the Jacobian matrix of $c(x)$ evaluated at x_k. Let l in NP be partitioned into three sections: the first n components (denoted by l_B), corresponding to the *bound* constraints; the next m_L components (denoted by l_L), corresponding to the *linear* constraints; and the last m_N components (denoted by l_N), corresponding to the *nonlinear constraints*. The vector $\bar{l}$ in QP is partitioned in the same way, and is defined as

$$\bar{l}_B = l_B - x_k, \quad \bar{l}_L = l_L - \mathcal{A}_L x_k, \quad \text{and} \quad \bar{l}_N = l_N - c_k,$$

where c_k is $c(x)$ evaluated at x_k. The vector $\bar{u}$ is defined in an analogous fashion.

In general, solving the subproblem QP for p_k is itself an iterative procedure. Hence, a "minor iteration" of NPSOL corresponds to an iteration within the QP algorithm. Note that the functions $F(x)$ and $c(x)$ are not evaluated during the solution of the subproblem. The total number of function evaluations required to solve a well-behaved problem will usually be similar to the number of major iterations.

The problem QP is solved using subroutines from the SOL/QPSOL package, which is described in detail in Gill *et al.* (1983a), and was specifically designed to be used within an SQP algorithm for nonlinear programming. In particular, two common difficulties associated with SQP methods are alleviated by certain features of the QPSOL subroutines.

First, it may happen that the QP subproblem is infeasible, yet feasible points exist with respect to the nonlinear constraints. (Throughout this report, we assume that "feasibility" is defined by a set of tolerances provided by the user in the array FEATOL; see Section 4.) The strategy used by NPSOL to treat an infeasible subproblem is the following. If there is no feasible point with respect to the bounds and linear constraints of the original problem, the infeasibility is inherent in the problem, and hence NPSOL terminates. Otherwise, the infeasibility results from the linearized nonlinear constraints; the least infeasible point is then computed, the appropriate constraint bounds are (temporarily) relaxed, and a relaxed quadratic program is solved for p_k.

Second, it is useful in an SQP algorithm to be able to use the prediction of the active set from each QP subproblem to solve the next subproblem more efficiently. This benefit is achieved in NPSOL by a "hot start" feature that allows the initial working set and part of its factorization to be specified. Within NPSOL, the prediction of the active set from one QP subproblem is used as the "hot start" estimate of the working set for the next QP. In practice, this means that the QP subproblems near the solution reach optimality in only one iteration. Furthermore, separate treatment of linear constraints means that it is usually possible to save work in performing the factorization of the working set at the beginning of the QP (since the rows of $\mathcal{A}$ corresponding to the linear constraints are unchanged).

The algorithm used in NPSOL will be discussed in a forthcoming report. Details of the algorithm of QPSOL are given in Gill *et al.* (1983b).

3. SPECIFICATION

```
SUBROUTINE NPSOL ( ITMAX, MSGLVL, N,
                   NCLIN, NCNLN, NCTOTL, NROWA, NROWJ, NROWR,
                   BIGBND, EPSAF, ETA, FTOL,
                   A, BL, BU, FEATOL,
                   CONFUN, OBJFUN, COLD, FEALIN, ORTHOG,
                   INFORM, ITER, ISTATE,
                   C, CJAC, CLAMDA, OBJF, OBJGRD, R, X,
                   IW, LENIW, W, LENW )

EXTERNAL           CONFUN, OBJFUN
LOGICAL            COLD, FEALIN, ORTHOG
INTEGER            ITMAX, MSGLVL, N, NCLIN, NCNLN, NCTOTL,
                   NROWA, NROWJ, NROWR, INFORM, ITER, LENIW, LENW
INTEGER            ISTATE(NCTOTL), IW(LENIW)
REAL               BIGBND, EPSAF, ETA, FTOL, OBJF
REAL               A(NROWA,N), BL(NCTOTL), BU(NCTOTL), FEATOL(NCTOTL),
                   C(NROWJ), CJAC(NROWJ,N), CLAMDA(NCTOTL),
                   OBJGRD(N), R(NROWR,N), X(N), W(LENW)
```

Note: Here and elsewhere, the specification of a parameter as REAL should be interpreted as *working precision*, which may be DOUBLE PRECISION in some circumstances.

4. INPUT PARAMETERS

ITMAX is an upper bound on the number of major iterations to be performed. Unless the problem is known to be exceptionally difficult, a sensible initial choice for ITMAX is 50.

MSGLVL indicates the amount of intermediate output desired (see Section 9 for a description of the printout). All output is written to the file number NOUT (see subroutine MCHPAR in Section 11). MSGLVL is interpreted as a four-digit number. Its first two digits indicate the level of intermediate output from the quadratic programming routines; the second two digits indicate the level of intermediate output from NPSOL. The QP printout levels are defined in Gill *et al.* (1983a); if MSGLVL < 100, there is no QP output. *When the last two digits of* MSGLVL ≥ 10, *each level includes the printout from all lower levels.* The printout corresponding to each value of the last two digits of MSGLVL is defined as follows:

Value	Definition
0	No output.
1	The final solution only.
5	One brief line of output for each major iteration (no printout of the final solution).
≥ 10	The final solution and one brief line of output for each major iteration.
≥ 15	At each iteration, the arrays X and ISTATE, and the indices of the free variables.
≥ 20	At each iteration, the nonlinear constraint values (the array C), the linear constraint values ($A_L x$), and estimates of the Lagrange multipliers.
≥ 30	At each iteration, the diagonal elements of the matrix T associated with the TQ factorization of the working set, and the diagonals of the matrix R (the Cholesky factor of the Hessian approximation).
≥ 80	Debug output from NPSOL.
99	Debug output from the line search.

For example, MSGLVL $= 10$ will produce a summary of results for each major iteration and a full printout of the final solution; MSGLVL $= 510$ will produce the same printout, as well as a summary of each minor (QP) iteration.

N is the number of variables, i.e., the dimension of X (N must be positive).

NCLIN is the number of general linear constraints in the problem (**NCLIN** may be zero).

NCNLN is the number of nonlinear constraints in the problem (**NCNLN** may be zero).

NCTOTL must be set to **N** + **NCLIN** + **NCNLN**.

NROWA is the declared row dimension of the array **A** (**NROWA** must be at least 1 and at least **NCLIN**).

NROWJ is the declared row dimension of the array **CJAC** and the length of the array **C** (**NROWJ** must be at least 1 and at least **NCNLN**).

NROWR is the declared row dimension of the array **R** (**NROWR** must be at least **N**).

BIGBND is a positive real variable whose magnitude denotes an "infinite" component of ℓ and u. Any upper bound greater than or equal to **BIGBND** will be regarded as plus infinity (and similarly for a lower bound less than or equal to −**BIGBND**).

EPSAF is a positive quantity that should be a good bound on the *absolute* error in computing $F(x)$ at the initial point. For many simple functions, **EPSAF** is of the order of $\epsilon_M|F(x)|$, where ϵ_M is the machine precision. A discussion of **EPSAF** is given in Chapter 8 of Gill, Murray and Wright (1981).

ETA is a number satisfying $0 \leq$ **ETA** < 1, which controls how accurately the value α_k approximates a univariate minimum of the merit function along p_k (the smaller the value of **ETA**, the more accurate the line search). The recommended value of **ETA** for nonlinearly constrained problems is 0.9, which corresponds to a relaxed line search. If the problem is unconstrained, bound-constrained, or linearly constrained, a smaller value of **ETA** will tend to require more function evaluations, but fewer major iterations.

FTOL is a positive tolerance (**FTOL** < 1) that indicates the number of figures of accuracy desired in the objective function at the solution. For example, if **FTOL** is 10^{-6} and **NPSOL** terminates successfully, the computed solution should have approximately six correct figures in F. **FTOL** should never be less than machine precision.

A is a real array of declared dimension (**NROWA**, **N**), corresponding to A_L in the problem specification NP (Section 1). The i-th row of **A**, $i = 1$ to **NCLIN**, contains the coefficients of the i-th general *linear* constraint. If **NCLIN** is zero, **A** is not accessed.

BL is a real array of dimension **NCTOTL** that contains the lower bounds for all the constraints, in the following order (which is also observed for **BU**, **CLAMDA**, **FEATOL** and **ISTATE**). The first **N** elements of **BL** contain the lower bounds on the variables. If **NCLIN** > 0, the next **NCLIN** elements of **BL** contain the lower bounds for the general linear constraints. If **NCNLN** > 0, the next **NCNLN** elements of **BL** contain the lower bounds for the nonlinear constraints. In order for the problem specification to be meaningful, it is required that $\text{BL}(j) \leq \text{BU}(j)$ for all j. To specify a non-existent lower bound for the j-th constraint (i.e., $\ell_j = -\infty$), the value used must satisfy $\text{BL}(j) \leq -\text{BIGBND}$. To specify the j-th constraint as an *equality*, the user must set $\text{BL}(j) = \text{BU}(j) = \beta$, say where $|\beta| <$ **BIGBND**.

BU is a real array of dimension **NCTOTL** that contains the upper bounds for all the constraints, in the same order described above for **BL**. To specify a non-existent upper bound (i.e., $u_j = +\infty$), the value used must satisfy $\text{BU}(j) \geq \text{BIGBND}$.

FEATOL is a real array of dimension **NCTOTL** containing positive tolerances that define the maximum permissible violation in each constraint in order for a point to be considered feasible, i.e. constraint j is considered satisfied if its violation does not exceed **FEATOL**(j). The ordering of the components of **FEATOL** is the same as that described above under **BL**. Note that **FEATOL**(j) is a bound on the *absolute* acceptable violation. For example, if the data defining the constraints are of order unity and are correct to about 6 decimal digits, it would be appropriate to choose **FEATOL**(j) as 10^{-6} for all relevant j. In general, the elements of **FEATOL** should be chosen as the *largest* possible acceptable values, since the algorithm of NPSOL becomes less likely to encounter difficulties with ill-conditioning and degeneracy as the components of **FEATOL** increase. A warning message is printed if any component of **FEATOL** is less than machine precision; the user must *not* set any element of **FEATOL** to zero. A detailed discussion of **FEATOL** is given in Gill *et al.* (1983b).

CONFUN is the name of a subroutine that calculates the vector $c(x)$ of nonlinear constraint functions and its Jacobian for a specified n-vector x. **CONFUN** must be declared as **EXTERNAL** in the routine that calls **NPSOL**. If there are no nonlinear constraints (**NCNLN** $= 0$), **CONFUN** will never be called by **NPSOL**. If there are nonlinear constraints, **NPSOL** always calls **CONFUN** and **OBJFUN** together, in that order.

The specification of **CONFUN** is:

```
SUBROUTINE CONFUN( MODE, NCNLN, N, NROWJ, X, C, CJAC, NSTATE )
INTEGER            MODE, NCNLN, N, NROWJ, NSTATE
REAL               X(N), C(NROWJ), CJAC(NROWJ,N).
```

The actual parameters **NCNLN**, **N**, and **NROWJ** input to **CONFUN** will always be the same Fortran variables as those input to **NPSOL**. They must *not* be altered by **CONFUN**.

MODE is a flag that the user may set within CONFUN to indicate a failure in the evaluation of the nonlinear constraints. On entry to CONFUN, MODE is always nonnegative. If MODE is negative on exit from CONFUN, the execution of NPSOL will be terminated with INFORM set to MODE.

X contains the vector of variables x at which the constraint functions are to be evaluated. The elements of X must *not* be altered by CONFUN.

C should contain the nonlinear constraint values $c_i(x)$, $i = 1$ to NCNLN, on exit from CONFUN.

CJAC should contain the Jacobian matrix of the nonlinear constraint functions on exit from CONFUN. The i-th row of CJAC contains the gradient of the i-th nonlinear constraint, i.e. CJAC(i, j) is the partial derivative of c_i with respect to x_j, $i = 1$ to NCNLN, $j = 1$ to N. *If* CJAC *contains any constant elements, a saving in computation can be made by setting them one time only, when* NSTATE $= 1$ (see below).

NSTATE is set to one by NPSOL on the first call of CONFUN, and is zero for all subsequent calls. Thus, if the user wishes, NSTATE may be tested within CONFUN in order to perform certain calculations one time only. For example, the user may read data or initialize COMMON blocks when NSTATE $= 1$. In addition, the constant elements of CJAC can be set in CONFUN when NSTATE $= 1$, and need not be defined on subsequent calls.

OBJFUN is the name of a subroutine that calculates the objective function $F(x)$ and its gradient for a specified n-vector x. OBJFUN must be declared as EXTERNAL in the routine that NPSOL.

The specification of OBJFUN is:

```
SUBROUTINE OBJFUN( MODE, N, X, OBJF, OBJGRD, NSTATE )
INTEGER            MODE, N, NSTATE
REAL               OBJF, X(N), OBJGRD(N).
```

The actual parameter N input to OBJFUN will always be the same Fortran variable as that input to NPSOL, and must *not* be altered by OBJFUN.

MODE is a flag that the user may set within OBJFUN to indicate a failure in the evaluation of the objective function. On entry to OBJFUN, MODE is always nonnegative. If MODE is negative on exit from OBJFUN, the execution of NPSOL is terminated with INFORM set to MODE.

X contains the vector of variables x at which the objective function is to be evaluated. The X array must *not* be altered by OBJFUN.

OBJF should contain the value of the objective function $F(x)$ on exit from OBJFUN.

OBJGRD should contain the gradient vector of the objective function. The j-th component of OBJGRD contains the partial derivative of F with respect to the j-th variable.

NSTATE is set to one by NPSOL on the first call of OBJFUN, and to zero on all subsequent calls. Thus, if the user wishes, NSTATE may be tested in order to perform certain calculations only on the first call of OBJFUN — e.g., read data or initialize COMMON blocks. Note that if there are any nonlinear constraints, CONFUN and OBJFUN are always called together, in that order.

COLD is a logical variable that indicates whether the user has specified an initial estimate of the active set of constraints. If COLD is .TRUE., the initial working set is determined by the first QP subproblem. If COLD is .FALSE. (a "warm start"), the user must define the array ISTATE (which gives the status of each constraint with respect to the working set) and the matrix R (the Cholesky factor of the initial Hessian approximation). The warm start option is particularly useful when NPSOL is restarted at the point where an earlier run terminated.

FEALIN is a logical variable that indicates whether the starting point for the SQP method should first be made feasible with respect to the bounds and linear constraints of NP. If FEALIN is .TRUE., the algorithm will determine (if possible) a point that is feasible with respect to the bounds and linear constraints before beginning the SQP iterations (where "feasible" is defined by the array FEATOL; see above). This setting of FEALIN ensures that *all* iterates within the SQP algorithm will be feasible with respect to the bounds and linear constraints (this may be essential in certain applications). If FEALIN is .FALSE., the SQP method will begin with the user-specified initial value of X. In this case, the iterates will not necessarily be feasible with respect to the linear constraints of the original problem (unless the original point is feasible). In general, we recommend a value of .TRUE. for FEALIN.

ORTHOG is a logical variable that indicates whether orthogonal transformations will be used in the QP algorithm to compute and update the TQ factorization of the working set

$$AQ = (0 \quad T),$$

where A is a submatrix of $\mathcal{A}$ and T is reverse-triangular (see Gill *et al.*, 1982). If ORTHOG is .TRUE., the TQ factorization is computed using Householder reflections and plane rotations, and the matrix Q is orthogonal. If ORTHOG is .FALSE., stabilized elementary transformations are used to maintain the factorization, and Q is not orthogonal. A rule of thumb in making the choice is that orthogonal transformations require more work, but provide greater numerical stability. Thus, we recommend setting ORTHOG to .TRUE. in any of the following situations: the problem is reasonably small; the functions are highly nonlinear; the active set is ill-conditioned; or the time required to compute the TQ factorization is not significant compared to the evaluation of the problem functions.

Otherwise, setting ORTHOG to .FALSE. will often lead to a reduction in solution time with negligible loss of reliability.

5. INPUT/OUTPUT PARAMETERS

ISTATE is an integer array of dimension NCTOTL that indicates the status of every constraint with respect to the current prediction of the active set. The ordering of ISTATE is the same as that described above for BL, i.e., the first N components of ISTATE refer to the bounds on the variables, the next NCLIN components refer to the linear constraints, and the last NCNLN components refer to the nonlinear constraints. The significance of each possible value of ISTATE(j) is as follows:

ISTATE(j)	Meaning
−2	This constraint (or its linearization) violates its lower bound by more than FEATOL(j) in a QP subproblem.
−1	This constraint (or its linearization) violates its upper bound by more than FEATOL(j) in a QP subproblem.
0	The constraint is not in the predicted active set.
1	This inequality constraint is included in the predicted active set at its lower bound.
2	This inequality constraint is included in the predicted active set at its upper bound.
3	The constraint is included in the predicted active set as an equality. This value of ISTATE can occur only when BL(j) = BU(j).

If COLD = .TRUE., ISTATE need not be set by the user. However, when COLD is .FALSE., every element of ISTATE must be set to one of the values given above to define a *suggested* prediction of the active set (which will be used as the initial working set in the first QP subproblem). The most likely values are:

ISTATE(j)	Meaning
0	The corresponding constraint should not be in the initial working set.
1	The constraint should be in the initial working set at its lower bound.
2	The constraint should be in the initial working set at its upper bound.
3	The constraint should be in the initial working set as an equality. This value must not be specified unless BL(j) = BU(j). The input values 1, 2 or 3 of ISTATE(j) all have the same effect when BL(j) = BU(j).

On exit from NPSOL, the values in the ISTATE array indicate the composition of the active set of the final QP subproblem.

R is a real array of declared dimension (NROWR,N) that contains the upper-triangular Cholesky factor of the current approximation of the Hessian of the Lagrangian function. If COLD is .TRUE., the array R need not be initialized by the user. If COLD is .FALSE., R must contain an appropriate upper-triangular matrix.

X is a real array of dimension N that contains the current estimate of the solution. On entry to NPSOL, X must be defined; on exit from NPSOL, X contains the final estimate of the solution.

6. OUTPUT PARAMETERS

INFORM is an integer that indicates the result of NPSOL. (When MSGLVL > 0, a short description of INFORM is printed.) The possible values of INFORM are:

INFORM	Definition
< 0	The user has set MODE to this negative value in CONFUN or OBJFUN.
0	X satisfies the first-order optimality conditions, i.e., the projected gradient and the active constraint residuals are negligible, and the Lagrange multipliers indicate optimality.
1	No feasible point could be found for the linear constraints and bounds.
2	No improved point for the merit function could be found during the final line search.
3	The limit of ITMAX major iterations was reached.
4	Extremely small Lagrange multipliers could not be resolved.
5	A descent direction for the merit function could not be found.
9	An input parameter is invalid.

ITER is an integer that gives the number of major iterations performed.

C is a real array of dimension NROWJ that contains the values of the nonlinear constraint functions C(i), $i = 1$ to NCNLN, at the final iterate. If NCNLN $= 0$, C is not accessed by NPSOL.

CJAC is a real array of dimension (NROWJ,N) that contains the Jacobian matrix of the nonlinear constraint functions at the final iterate, i.e. CJAC(i, j) contains the partial derivative of the i-th constraint function with respect to the j-th variable, $i = 1$ to NCNLN, $j = 1$ to N. If NCNLN $= 0$, CJAC is not accessed by NPSOL. (See the discussion of CJAC under CONFUN above.)

CLAMDA is a real array of dimension NCTOTL that contains the final multiplier estimate for every constraint (i.e., the multipliers of the final QP subproblem). The ordering of CLAMDA is the same as that given above for BL. If the j-th constraint is defined as "inactive" by the ISTATE array, CLAMDA(j) should be zero; if the j-th constraint is an inequality active at its lower bound, CLAMDA(j) should be non-negative; if the j-th constraint is an inequality active at its upper bound, CLAMDA(j) should be non-positive.

OBJF is the value of the objective function $F(x)$ at the final iterate.

OBJGRD is a real array of dimension N that contains the gradient of the objective function.

7. WORKSPACE PARAMETERS

IW is an integer array of dimension **LENIW**, which provides integer workspace for **NPSOL**.

LENIW is the dimension of **IW**, and must be at least **2N**.

W is a real array of dimension **LENW**, which provides real workspace for **NPSOL**.

LENW is the dimension of **W**, and must be at least $2N^2 + N(NCON + NROWJ + 6) + 2NCON + NROWA + \max(10N + 2NCON + NROWA + NROWJ, 5N + 4NCON)$, where $NCON = \max(1, NCLIN + NCNLN)$. An overestimate of this number is $2N^2 + N(NCON + NROWJ + 16) + 6NCON + 2NROWA + NROWJ$.

If **MSGLVL** > 0, the amount of workspace provided and the amount of workspace required are printed. As an alternative to computing **LENW** from the formula given above, the user may prefer to obtain an appropriate value from the output of a preliminary run with a positive value of **MSGLVL** and **LENW** set to 1 (**NPSOL** will then terminate with **INFORM** $= 9$).

8. AUXILIARY SUBPROGRAMS AND LABELLED COMMON

The auxiliary subroutines used by NPSOL may be divided into three groups. The first group includes the following subroutines, which are not part of the QP package:

GETPTC	NPCORE	NPGETC	NPGETF
NPGLF	NPHESS	NPIQP	NPPRT
NPQPGN	NPRHO	NPSRCH	NPTQ
R1BFGS	R1MOD.		

The second group of subroutines — those used by the QP package — are:

ADDCON	ALLOC	BDPERT	BNDALF
CHKDAT	DELCON	FINDP	GETLAM
LPBGST	LPCORE	LPCRSH	LPDUMP
LPGRAD	LPPRT	MOVEX	QPCHKP
QPCOLR	QPCORE	QPCRSH	QPDUMP
QPGRAD	QPPRT	PRTSOL	RSOLVE
TQADD	TSOLVE	ZYPROD.	

NPSOL also uses the basic linear algebra subroutines

AXPY	CONDVC	COPYMX	COPYVC
DOT	DSCALE	ELM	ELMGEN
ETAGEN	QUOTNT	REFGEN	ROT3
ROTGEN	SCMOVE	V2NORM	ZEROVC

and the subroutine MCHPAR, which defines machine-dependent constants (see Section 11).

The subroutines in the NPSOL package use the following labelled COMMON areas:

SOLMCH (15 REAL variables; see Section 11)
SOL1CM (3 INTEGER variables)
SOL3CM (4 INTEGER variables)
SOL4CM (10 REAL variables)
SOL1LP (15 INTEGER variables)
SOL1NP (30 INTEGER variables)
SOL2NP (2 INTEGER variables).

9. DESCRIPTION OF THE PRINTED OUTPUT

The following is a description of the terse line printed at each major iteration if the last two digits of MSGLVL ≥ 5. The printout from the QP subroutines is described in Gill *et al.* (1983a). All quantities are evaluated at the end of the iteration.

ITN	is the major iteration count, k.
ITQP	is the number of minor iterations needed to solve the QP subproblem.
STEP	is the step α_k taken along the computed search direction.
NUMF	is the total number of evaluations of the problem functions.
OBJECTIVE	is the value of the objective function, $F(x_k)$.
BND	is the number of bounds in the predicted active set.
LC	is the number of linear constraints in the predicted active set.
NC	is the number of nonlinear constraints in the predicted active set.
NCOLZ	is N minus the number of constraints in the predicted active set.
NORM GFREE	is the norm of the gradient of the objective function with respect to the free variables (not printed if ORTHOG is .FALSE.).
NORM QTG	is a weighted norm of the gradient of the objective function with respect to the free variables (not printed if ORTHOG is .TRUE.).
NORM ZTG	is the Euclidean norm of the projected gradient.
COND H	is a lower bound on the condition number of the Hessian approximation, i.e. a bound on $\text{cond}(H) = \text{cond}(R^T R)$.
COND T	is a lower bound on the condition number of the matrix of predicted active constraints.
NORM C	is the norm of the vector of constraint violations and residuals of the constraints in the predicted active set.
RHO	is the penalty parameter used in the augmented Lagrangian merit function.

CONV	is a four-letter indication of the status of the four convergence tests; each letter is "T" if the test is satisfied, and "F" otherwise. The four tests indicate whether: (a) the projected gradient is small; (b) the active constraint residuals are small; (c) the multipliers indicate optimality; (d) the last change in X was small.
U	refers to the quasi-Newton update of R to obtain a new estimate of the Hessian. U is 1 if the update was performed, and 0 if no update occurred.

The following is a description of the *solution output* of NPSOL. Note that names are automatically assigned to each variable and constraint.

The following printout is given for each variable x_j.

VARIABLE	is the name (VARBL) and index j, $j = 1$ to N, of the variable.
STATE	gives the state of the variable (FR if not in the working set, EQ if in the working set as a fixed variable, LL if in the working set at its lower bound, and UL if in the working set at its upper bound). If VALUE lies outside the upper or lower bounds by more than FEATOL(j), STATE will be "++" or "--" respectively.
VALUE	is the value of the variable x_j at the final iteration.
LOWER BOUND	is the lower bound BL(j) specified for the variable.
UPPER BOUND	is the upper bound BU(j) specified for the variable.
LAGR MULTIPLIER	is the value of the Lagrange multiplier for the corresponding bound constraint. This will be zero if STATE is FR. If X is optimal and STATE is LL, the multiplier should be non-negative; if STATE is UL, the multiplier should be non-positive.
RESIDUAL	is the difference between the variable and its nearer bound.

The following printout is given for each constraint.

LINEAR CONSTR	is the name (LNCON) and index i, $i = 1$ to NCLIN, of a linear constraint.
NONLNR CONSTR	is the name (NLCON) and index i, $i = 1$ to NCNLN, of a nonlinear constraint.

STATE is the state of the constraint (FR for a constraint not in the working set, EQ for an equality in the working set, LL for an inequality constraint in the working set at its lower bound, UL for an inequality constraint in the working set at its upper bound). STATE will be "++" or "--" respectively if VALUE lies outside the upper or lower bounds by more than its feasibility tolerance.

VALUE is the value of the constraint at the final point.

LOWER BOUND is the specified lower bound for the constraint.

UPPER BOUND is the specified upper bound for the constraint.

LAGR MULTIPLIER is the value of the Lagrange multiplier. This will be zero if STATE is FR. If X is optimal and STATE is LL, the multiplier should be non-negative; if STATE is UL, the multiplier should be non-positive.

RESIDUAL is the residual of the constraint with respect to its nearer bound, i.e., the difference between VALUE and the nearer of the two bounds.

10. ERROR RECOVERY

The input data for NPSOL should always be checked (even if NPSOL terminates with the value INFORM = 0!). Two common sources of error are uninitialized variables and incorrect gradients, which may cause underflow or overflow on some machines. The user should check that all components of A, BL, BU, FEATOL and X are defined on entry to NPSOL, and that OBJFUN and CONFUN compute all relevant components of OBJGRD, C and CJAC.

The present version of NPSOL contains no procedure for checking the computed gradients. Incorrect gradients may lead to termination with INFORM = 2, 3 or 5.

Other error conditions may arise as follows.

Termination	Recommended Action
Underflow	If the machine parameter indicating an underflow check (WMACH(9)) is zero, floating-point underflow may occur occasionally, but can usually be ignored. To avoid underflow, set WMACH(9) to a positive value; however, this will lead to a noticeable loss of efficiency. If underflow continues to occur for no apparent reason, contact the authors at Stanford University.
Overflow	If the printed output before the overflow error contains a warning about serious ill-conditioning in the working set when adding the j-th constraint, it may be possible to avoid the difficulty by increasing the magnitude of FEATOL(j), and rerunning the program. If the message recurs even after this change, the offending linearly dependent constraint must be removed from the problem. If overflow occurs in one of the user-supplied routines (e.g., if the nonlinear functions involve exponentials or singularities), it may help to specify tighter bounds for some of the variables (i.e., reduce the gap between appropriate ℓ_j and u_j). If overflow continues to occur for no apparent reason, contact the authors at Stanford University.
INFORM = 1	A feasible point could not be found for the bounds and linear constraints. This exit occurs if there is a failure in the LP phase of any QP subproblem (see Gill *et al.*, 1983a). The most likely reason for this condition is that the linear constraints and bounds are incompatible or inconsistent; if so, NPSOL will terminate during the first major iteration. In order for a feasible point to exist, the constraints must be re-formulated, or the corresponding components of FEATOL must be re-defined, as discussed in Gill *et al.* (1983a). Another possibility is that dependencies among the constraints and bounds have led to cycling in the LP phase; this will

always be the case if **NPSOL** terminates with **INFORM** $= 1$ after the first major iteration.

INFORM $= 2$ A sufficient decrease in the merit function could not be attained during the final line search. This sometimes occurs because an overly stringent accuracy has been requested, i.e., **FTOL** is too small; in this case the final solution may be acceptable despite the non-zero value of **INFORM** (see Gill, Murray and Wright, 1981, for a discussion of the attainable accuracy). If the projected gradient at the final point is not small, the computed gradients may be incorrect. Another possibility is that the search direction has become inaccurate because of ill-conditioning in the Hessian approximation or the matrix of constraints in the working set; either form of ill-conditioning also tends to be reflected in large values of **ITQP** (the number of iterations required to solve each QP subproblem). If the condition estimate of the Hessian (**COND H**) is extremely large, it may be worthwhile to try a warm start at the final point with **COLD** set to **.FALSE.**, **ISTATE** unaltered, and **R** set to the identity matrix. If the matrix of constraints in the working set is ill-conditioned (i.e., **COND T** is extremely large), it may be helpful to run **NPSOL** with relaxed values of the components of **FEATOL** corresponding to nearly dependent constraints. (Constraint dependencies are often indicated by wide variations in size in the diagonal elements of the matrix T, whose diagonals will be printed if the last two digits of **MSGLVL** ≥ 30.)

INFORM $= 3$ If the algorithm appears to be making progress, the value of **ITMAX** may be too small. If so, increase **ITMAX** and rerun **NPSOL** (possibly using the warm start facility). If the algorithm seems to be "bogged down", the user should check for incorrect gradients or ill-conditioning as described above under **INFORM** $= 2$. Note that ill-conditioning in the working set is sometimes resolved automatically by the algorithm, in which case performing additional iterations may be helpful. However, ill-conditioning in the Hessian approximation tends to persist once it has begun, so that allowing additional iterations without altering **R** is usually inadvisable. If the constraint violations have not been significantly reduced, the problem may have no feasible point.

INFORM $= 4$ A guaranteed procedure for resolving extremely small Lagrange multipliers has not been included in **NPSOL**, since it would be inherently combinatorial (see Gill, Murray and Wright, 1981, for further discussion). In some cases, the difficulty may be avoided by removing certain active constraints with very small multipliers from the problem, and rerunning `NPSOL`.

`INFORM` = 5 With exact arithmetic, the search direction should always be a descent direction for the merit function. If this value of `INFORM` occurs, the computed gradients may be incorrect, or ill-conditioning may have destroyed the accuracy of the search direction. The user should check for these conditions as described above under `INFORM` = 2.

11. IMPLEMENTATION INFORMATION

This program has been written in ANSI (1966) Fortran and tested on an IBM 3081 computer using the WATFIV Compiler, Version 1, Level 6. All subroutines in NPSOL are PFORT-compatible (Ryder, 1974), except for some A2 Hollerith specifications.

At the beginning of NPSOL, the subprogram MCHPAR is called to assign various machine-dependent parameters. These parameters are stored in the array WMACH(15) in the labelled COMMON block SOLMCH.

The specification of MCHPAR is

```
SUBROUTINE MCHPAR
REAL                 WMACH
COMMON     /SOLMCH/  WMACH(15)
```

The first eleven components of the REAL array WMACH must be set in MCHPAR. The components of WMACH are defined as follows.

	Definition
WMACH(1)	is NBASE, the base of floating-point arithmetic.
WMACH(2)	is NDIGIT, the number of NBASE digits of precision.
WMACH(3)	is EPSMCH, the floating-point precision.
WMACH(4)	is RTEPS, the square root of EPSMCH.
WMACH(5)	is FLMIN, the smallest positive floating-point number.
WMACH(6)	is RTMIN, the square root of FLMIN.
WMACH(7)	is FLMAX, the largest positive floating-point number.
WMACH(8)	is RTMAX, the square root of FLMAX.
WMACH(9)	is UNDFLW, which specifies whether or not NPSOL should check for underflow in certain computations. If UNDFLW $=$ 0, no underflow checking will be performed. If UNDFLW is set to a positive number, NPSOL will check for underflow and will replace too-small quantities by zero. *Note that NPSOL will run faster if no underflow checking takes place.*
WMACH(10)	is NIN, the file number for the input stream.
WMACH(11)	is NOUT, the file number for the output stream.

The following version of MCHPAR (which is provided by the Systems Optimization Laboratory) contains the parameters associated with double precision on a machine in the IBM 370 series. *The user must substitute a version of* MCHPAR *that is appropriate for the machine to be used.*

```
      SUBROUTINE MCHPAR
C
      DOUBLE PRECISION   WMACH
      COMMON    /SOLMCH/ WMACH(15)
C
C  MCHPAR  MUST DEFINE THE RELEVANT MACHINE PARAMETERS AS FOLLOWS.
C     WMACH(1)  = NBASE  = BASE OF FLOATING-POINT ARITHMETIC.
C     WMACH(2)  = NDIGIT = NO. OF BASE WMACH(1) DIGITS OF PRECISION.
C     WMACH(3)  = EPSMCH = FLOATING-POINT PRECISION.
C     WMACH(4)  = RTEPS  = SQRT(EPSMCH).
C     WMACH(5)  = FLMIN  = SMALLEST POSITIVE FLOATING-POINT NUMBER.
C     WMACH(6)  = RTMIN  = SQRT(FLMIN).
C     WMACH(7)  = FLMAX  = LARGEST POSITIVE FLOATING-POINT NUMBER.
C     WMACH(8)  = RTMAX  = SQRT(FLMAX).
C     WMACH(9)  = UNDFLW = 0.0 IF UNDERFLOW IS NOT FATAL, +VE OTHERWISE.
C     WMACH(10) = NIN    = STANDARD FILE NUMBER OF THE INPUT STREAM.
C     WMACH(11) = NOUT   = STANDARD FILE NUMBER OF THE OUTPUT STREAM.
C
      INTEGER            NBASE, NDIGIT, NIN, NOUT
      DOUBLE PRECISION   DSQRT
C
      NBASE     = 16
      NDIGIT    = 14
      WMACH(1)  = NBASE
      WMACH(2)  = NDIGIT
      WMACH(3)  = WMACH(1)**(1 - NDIGIT)
      WMACH(4)  = DSQRT(WMACH(3))
      WMACH(5)  = WMACH(1)**(-62)
      WMACH(6)  = DSQRT(WMACH(5))
      WMACH(7)  = WMACH(1)**61
      WMACH(8)  = DSQRT(WMACH(7))
      WMACH(9)  = 0.0D+0
      NIN       = 5
      NOUT      = 6
      WMACH(10) = NIN
      WMACH(11) = NOUT
C
C---- IN WATFIV, ALLOW UP TO 100 UNDERFLOWS.
C---- CALL TRAPS ( 0,0,100 )
      RETURN
C
C  END OF MCHPAR
      END
```

The values of NBASE, NDIGIT, EPSMCH, FLMIN and FLMAX for several machines are given in the following table, for both single and double precision; RTEPS, RTMIN and RTMAX may be computed using Fortran statements. The values NIN and NOUT depend on the machine installation.

For each precision, we give two values for EPSMCH, FLMIN and FLMAX. The first value is a Fortran decimal approximation of the exact quantity; use of this value in MCHPAR should cause no difficulty except in extreme circumstances. The second value is the exact mathematical representation.

Table of machine-dependent parameters

Variable	IBM 360/370 Single	CDC 6000/7000 Single	DEC 10/20 Single	Univac 1100 Single	DEC VAX Single
NBASE	16	2	2	2	2
NDIGIT	6	48	27	27	24
EPSMCH	9.54E-7 16^{-5}	7.11E-15 2^{-47}	7.46E-9 2^{-27}	1.50E-8 2^{-26}	1.20E-7 2^{-23}
FLMIN	1.0E-78 16^{-65}	1.0E-293 2^{-975}	1.0E-38 2^{-129}	1.0E-38 2^{-129}	1.0E-38 2^{-128}
FLMAX	1.0E+75 $16^{63}(1-16^{-6})$	1.0E+322 $2^{1070}(1-2^{-48})$	1.0E+38 $2^{127}(1-2^{-27})$	1.0E+38 $2^{127}(1-2^{-27})$	1.0E+38 $2^{127}(1-2^{-24})$

Variable	IBM 360/370 Double	CDC 6000/7000 Double	DEC 10/20 Double	Univac 1100 Double	DEC VAX Double
NBASE	16	2	2	2	2
NDIGIT	14	96	62	61	56
EPSMCH	2.22D-16 16^{-13}	2.53D-29 2^{-95}	2.17D-19 2^{-62}	8.68D-19 2^{-60}	2.78D-17 2^{-55}
FLMIN	1.0D-78 16^{-65}	1.0D-293 2^{-975}	1.0D-38 2^{-129}	1.0D-308 2^{-1025}	1.0D-38 2^{-128}
FLMAX	1.0D+75 $16^{63}(1-16^{-14})$	1.0D+322 $2^{1070}(1-2^{-96})$	1.0D+38 $2^{127}(1-2^{-62})$	1.0D+307 $2^{1023}(1-2^{-61})$	1.0D+38 $2^{127}(1-2^{-56})$

12. EXAMPLE PROGRAM AND OUTPUT

This section contains a listing and the computed results from a sample main program that calls NPSOL to solve one version of the so-called "hexagon" problem (a different formulation is given as Problem 108 in Hock and Schittkowski, 1981). The problem is to determine the hexagon of maximum area such that no two of its vertices are more than one unit apart (the solution is *not* a regular hexagon).

All constraint types are included (bounds, linear, nonlinear), and the Hessian of the Lagrangian function is not positive definite at the solution. The problem has nine variables, non-infinite bounds on six of the variables, four general linear constraints, and fifteen nonlinear constraints.

The objective function is

$$F(x) = -x_2x_6 + x_1x_7 - x_3x_7 - x_5x_8 + x_4x_9 + x_3x_8.$$

The bounds on the variables are

$$x_1 \geq 0, \quad x_5 \geq 0, \quad x_6 \geq 0, \quad x_7 \geq 0, \quad x_8 \leq 0, \quad \text{and} \quad x_9 \leq 0.$$

Thus,

$$\ell_B = (\ \ 0, -\infty, -\infty, -\infty, \ \ 0, \ \ 0, \ \ 0, -\infty, -\infty)^T$$
$$u_B = (+\infty, +\infty, +\infty, +\infty, +\infty, +\infty, +\infty, \ \ 0, \ \ 0)^T.$$

The general linear constraints are

$$x_2 - x_1 \geq 0, \quad x_3 - x_2 \geq 0, \quad x_3 - x_4 \geq 0, \quad \text{and} \quad x_4 - x_5 \geq 0.$$

Hence,

$$\ell_L = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad A_L = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \text{and} \quad u_L = \begin{pmatrix} +\infty \\ +\infty \\ +\infty \\ +\infty \end{pmatrix}.$$

The fifteen nonlinear constraint functions are

$$\begin{array}{lll} c_1(x) = x_1^2 + x_6^2, & c_2(x) = (x_2 - x_1)^2 + (x_7 - x_6)^2, & c_3(x) = (x_3 - x_1)^2 + x_6^2, \\ c_4(x) = (x_1 - x_4)^2 + (x_6 - x_8)^2, & c_5(x) = (x_1 - x_5)^2 + (x_6 - x_9)^2, & c_6(x) = x_2^2 + x_7^2, \\ c_7(x) = (x_3 - x_2)^2 + x_7^2, & c_8(x) = (x_4 - x_2)^2 + (x_8 - x_7)^2, & c_9(x) = (x_2 - x_5)^2 + (x_7 - x_9)^2, \\ c_{10}(x) = x_3^2, & c_{11}(x) = (x_4 - x_3)^2 + x_8^2, & c_{12}(x) = (x_5 - x_3)^2 + x_9^2, \\ c_{13}(x) = x_4^2 + x_8^2, & c_{14}(x) = (x_4 - x_5)^2 + (x_9 - x_8)^2, & c_{15}(x) = x_5^2 + x_9^2. \end{array}$$

(For most applications, it would be preferable to replace the tenth nonlinear constraint ($x_3^2 \leq 1$) by the bounds $-1 \leq x_3 \leq 1$.

The nonlinear constraints are all of the form

$$c_i(x) \leq 1, \quad i = 1, \ldots, 15;$$

hence, all components of ℓ_N are $-\infty$, and all components of u_N are 1.

The starting point x_0 is

$$x_0 = (\ .33,\ .67,\ 1.1,\ .67,\ .33,\ .33,\ .67,\ -.33,\ -.67\ )^T,$$

and $F(x_0) = -1.4333$ (to five figures). The optimal solution (to five figures) is

$$x^* = (\ .060947,\ .59765,\ 1.0,\ .59765,\ .060947,\ .34377,\ .5,\ -.5,\ -.34377\ )^T,$$

and $F(x^*) = -1.34996$. (The optimal objective function is unique, but is achieved for other values of x.) Six nonlinear constraints are active at x^*. The sample solution output is given later in this section, following the sample main program and problem definition.

```
C
C  EXAMPLE PROGRAM FOR SUBROUTINE NPSOL
C  DOUBLE PRECISION VERSION 1.1.  APRIL 1983.
C  THE VALUES OF THE PARAMETERS   EPSAF, FTOL,  AND  FEATOL  ARE
C  APPROPRIATE FOR A  MACHINE WITH A PRECISION OF 15  DECIMAL DIGITS.
C  ************************************************************************
      INTEGER            I, INFORM, ITER, ITMAX, J, LIWORK, LWORK
      INTEGER            MSGLVL, N, NCLIN, NCNLN, NCTOTL
      INTEGER            NOUT, NROWA, NROWJ, NROWR, NSTATE
      INTEGER            ISTATE(28)
      INTEGER            IWORK(50)
      DOUBLE PRECISION   BIGBND, EPSAF, EPSMCH, RTEPS, ETA, FTOL, OBJF
      DOUBLE PRECISION   A(5,9), BL(28), BU(28), FEATOL(28)
      DOUBLE PRECISION   C(20), CJAC(20,9), CLAMDA(28)
      DOUBLE PRECISION   OBJGRD(9), R(10,9), X(9)
      DOUBLE PRECISION   WORK(1000)
      DOUBLE PRECISION   DSQRT
      DOUBLE PRECISION   ZERO, ONE
      LOGICAL            COLD, FEALIN, ORTHOG
      EXTERNAL           OBJFUN, CONFUN
      DATA               ZERO   , ONE
     *                  /0.0D+0, 1.0D+0/
C
C  SET THE DECLARED ARRAY DIMENSIONS.
C  NROWA  = THE DECLARED ROW DIMENSION OF  A.
C  NROWJ  = THE DECLARED ROW DIMENSION OF  CJAC.
C  NROWR  = THE DECLARED ROW DIMENSION OF  R.
C  LIWORK = THE LENGTH OF THE INTEGER WORK ARRAY.
C  LWORK  = THE LENGTH OF THE DOUBLE PRECISION WORK ARRAY.
C
      NROWA  = 5
      NROWJ  = 20
      NROWR  = 10
      LIWORK = 50
      LWORK  = 1000
C
C  SET THE APPROXIMATE MACHINE PRECISION.
C
      EPSMCH = 1.0D-15
C
C  SET THE PROBLEM DIMENSIONS.
C  N      = THE NUMBER OF VARIABLES.
C  NCLIN  = THE NUMBER OF GENERAL LINEAR CONSTRAINTS (MAY BE 0).
C  NCNLN  = THE NUMBER OF NONLINEAR CONSTRAINTS (MAY BE 0).
C  NCTOTL = THE TOTAL NUMBER OF VARIABLES AND CONSTRAINTS.
C           (THE ARRAYS  ISTATE, BL, BU, CLAMBDA  MUST BE AT LEAST
C           THIS LONG.)
C
      N      = 9
      NCLIN  = 4
      NCNLN  = 15
      NCTOTL = N + NCLIN + NCNLN
C
C  ASSIGN THE DATA ARRAYS.
C  BOUNDS  .GE.    BIGBND  WILL BE TREATED AS PLUS  INFINITY.
C  BOUNDS  .LE.  - BIGBND  WILL BE TREATED AS MINUS INFINITY.
C  NOUT    = THE UNIT NUMBER FOR PRINTING.
```

```
C  A      = THE GENERAL CONSTRAINT MATRIX.
C  BL     = THE LOWER BOUNDS ON  X,  A*X  AND  C(X).
C  BU     = THE UPPER BOUNDS ON  X,  A*X  AND  C(X).
C  X      = THE INITIAL ESTIMATE OF THE SOLUTION.
C
      NOUT   = 6
      BIGBND = 1.0D+10
      DO 30 J = 1, NCTOTL
         BL(J) = -BIGBND
         BU(J) =  BIGBND
   30 CONTINUE
      BL(1)  = ZERO
      BL(5)  = ZERO
      BL(6)  = ZERO
      BL(7)  = ZERO
C
C  SET LOWER BOUNDS OF ZERO FOR THE FOUR LINEAR CONSTRAINTS.
C
      BL(10) = ZERO
      BL(11) = ZERO
      BL(12) = ZERO
      BL(13) = ZERO
C
      BU(8)  = ZERO
      BU(9)  = ZERO
C
C  SET UPPER BOUNDS OF ONE FOR ALL 15 NONLINEAR CONSTRAINTS.
C
      DO 40 J = 14, 28
         BU(J) = ONE
   40 CONTINUE
C
      X(1)   =  .33D+0
      X(2)   =  .67D+0
      X(3)   =  1.1D+0
      X(4)   =  .67D+0
      X(5)   =  .33D+0
      X(6)   =  .33D+0
      X(7)   =  .67D+0
      X(8)   = -.33D+0
      X(9)   = -.67D+0
C
      DO 60 J = 1, N
         DO 50 I = 1, NCLIN
            A(I,J) = ZERO
   50    CONTINUE
   60 CONTINUE
      A(1,1) = -ONE
      A(1,2) =  ONE
      A(2,2) = -ONE
      A(2,3) =  ONE
      A(3,3) =  ONE
      A(3,4) = -ONE
      A(4,4) =  ONE
      A(4,5) = -ONE
C
C  PRINT THE DATA.
C
      WRITE (NOUT, 2100)
      DO 70 I = 1, NCLIN
```

```
            WRITE (NOUT, 2200) I, (A(I,J), J=1,N)
      70 CONTINUE
         WRITE (NOUT, 2300) (BL(J), J=1,NCTOTL)
         WRITE (NOUT, 2400) (BU(J), J=1,NCTOTL)
         WRITE (NOUT, 2500) ( X(J), J=1,N)
C
C
C     ALLOW UP TO 50 MAJOR ITERATIONS TO FIND A SOLUTION.
C
         ITMAX  = 50
C
C     ASK FOR BRIEF OUTPUT EACH MAJOR ITERATION, AND A FULL PRINT-OUT OF
C     THE FINAL SOLUTION.
C
         MSGLVL = 10
C
C     SET THE ABSOLUTE PRECISION OF THE OBJECTIVE AT THE STARTING POINT.
C
         NSTATE = 1
         CALL OBJFUN( 2, N, X, OBJF, OBJGRD, NSTATE )
         EPSAF  = EPSMCH * DABS( OBJF )
C
C     USE A SLACK LINESEARCH.
C     SET THE REQUIRED NUMBER OF CORRECT FIGURES IN THE OPTIMAL OBJECTIVE.
C     THE VALUE CHOSEN HERE  (FTOL = 10 EPSMCH)  ASKS FOR ALMOST FULL
C     PRECISION IN  OBJF.
C
         ETA    = 0.9D+0
         FTOL   = 10.0D+0 * EPSMCH
C
C     AT THE SOLUTION,  ANY CONSTRAINT MAY BE VIOLATED BY AS MUCH AS
C     THE SQUARE ROOT OF THE MACHINE PRECISION.
C
         RTEPS  = DSQRT( EPSMCH )
         DO 80 J = 1, NCTOTL
            FEATOL(J) = RTEPS
      80 CONTINUE
C
C     A COLD START IS NEEDED FOR THE FIRST CALL TO  NPSOL.
C     START THE NONLINEAR ITERATIONS AT A POINT THAT IS FEASIBLE WITH
C     RESPECT TO THE LINEAR CONSTRAINTS AND BOUNDS.
C     USE AN ORTHOGONAL FACTORIZATION OF THE MATRIX OF CONSTRAINTS
C     IN THE WORKING SET.
C
         COLD   = .TRUE.
         FEALIN = .TRUE.
         ORTHOG = .TRUE.
C
C
C     SOLVE THE PROBLEM.
C
         CALL NPSOL( ITMAX, MSGLVL, N,
     *               NCLIN, NCNLN, NCTOTL, NROWA, NROWJ, NROWR,
     *               BIGBND, EPSAF, ETA, FTOL,
     *               A, BL, BU, FEATOL,
     *               CONFUN, OBJFUN, COLD, FEALIN, ORTHOG,
     *               INFORM, ITER, ISTATE,
     *               C, CJAC, CLAMDA, OBJF, OBJGRD, R, X,
     *               IWORK, LIWORK, WORK, LWORK )
C
```

```
C  TEST FOR AN ERROR CONDITION.
C
      IF (INFORM .GT. 0) GO TO 900
C
C
C  THE FOLLOWING IS FOR ILLUSTRATIVE PURPOSES ONLY.
C  WE DO A WARM START WITH THE FINAL WORKING SET AND  R  OF THE PREVIOUS
C  RUN, BUT WITH A SLIGHTLY PERTURBED STARTING POINT.
C
      DO 100 J = 1, N
         X(J) = X(J) + 0.05D+0
  100 CONTINUE
C
C  RESET THE ABSOLUTE PRECISION OF THE OBJECTIVE FUNCTION.
C
      EPSAF  = EPSMCH * DABS( OBJF )
C
      COLD   = .FALSE.
      MSGLVL = 5
      WRITE (NOUT, 2600)
      WRITE (NOUT, 2500) (X(J), J=1,N)
C
      CALL NPSOL( ITMAX, MSGLVL, N,
     *            NCLIN, NCNLN, NCTOTL, NROWA, NROWJ, NROWR,
     *            BIGBND, EPSAF, ETA, FTOL,
     *            A, BL, BU, FEATOL,
     *            CONFUN, OBJFUN, COLD, FEALIN, ORTHOG,
     *            INFORM, ITER, ISTATE,
     *            C, CJAC, CLAMDA, OBJF, OBJGRD, R, X,
     *            IWORK, LIWORK, WORK, LWORK )
C
      IF (INFORM .GT. 0) GO TO 900
      STOP
C
C  ERROR EXIT.
C
  900 WRITE (NOUT, 3000) INFORM
      STOP
C
 2100 FORMAT(/ 12H ROWS OF  A.)
 2200 FORMAT(/ (1X, I3, 4X, 9F8.2))
 2300 FORMAT(/ 14H LOWER BOUNDS. / (1X, 1P7E10.2))
 2400 FORMAT(/ 14H UPPER BOUNDS. / (1X, 1P7E10.2))
 2500 FORMAT(/ 12H INITIAL  X.   / (1X,   7F10.2))
 2600 FORMAT(//48H A RUN OF THE SAME EXAMPLE WITH A WARM START....)
 3000 FORMAT(/ 32H NPSOL TERMINATED WITH  INFORM =, I3)
C
C  END OF THE EXAMPLE PROGRAM FOR NPSOL.
      END

      SUBROUTINE OBJFUN( MODE, N, X, OBJF, OBJGRD, NSTATE )
      INTEGER            MODE, N, NSTATE
      DOUBLE PRECISION   OBJF
      DOUBLE PRECISION   X(N), OBJGRD(N)
C
C  ---------------------------------------------------------------------
C  OBJFUN  COMPUTES THE VALUE AND FIRST DERIVATIVES OF THE NONLINEAR
C  OBJECTIVE FUNCTION.
C  ---------------------------------------------------------------------
      OBJF   = - X(2)*X(6) + X(1)*X(7) - X(3)*X(7) - X(5)*X(8)
     *         + X(4)*X(9) + X(3)*X(8)
```

```
C
      OBJGRD(1) =   X(7)
      OBJGRD(2) = - X(6)
      OBJGRD(3) = - X(7) + X(8)
      OBJGRD(4) =   X(9)
      OBJGRD(5) = - X(8)
      OBJGRD(6) = - X(2)
      OBJGRD(7) = - X(3) + X(1)
      OBJGRD(8) = - X(5) + X(3)
      OBJGRD(9) =   X(4)
      RETURN
C
C  END OF OBJFUN
      END

      SUBROUTINE CONFUN( MODE, NCNLN, N, NROWJ, X, C, CJAC, NSTATE )
      INTEGER            MODE, NCNLN, N, NROWJ, NSTATE
      DOUBLE PRECISION   X(N), C(NROWJ), CJAC(NROWJ,N)
C
C  ---------------------------------------------------------------
C  CONFUN  COMPUTES THE VALUES AND FIRST DERIVATIVES OF THE NONLINEAR
C  CONSTRAINTS.
C
C  THE ZERO ELEMENTS OF JACOBIAN MATRIX ARE SET ONLY ONCE.  THIS OCCURS
C  DURING THE FIRST CALL TO CONFUN  (NSTATE = 1).
C  ---------------------------------------------------------------
      INTEGER            I, J
      DOUBLE PRECISION   ZERO, TWO
      DATA               ZERO  , TWO
     *                  /0.0D+0, 2.0D+0/
C
      IF (NSTATE .NE. 1) GO TO 200
      DO 120 J = 1, N
         DO 110 I = 1, NCNLN
            CJAC(I,J) = ZERO
  110    CONTINUE
  120 CONTINUE
C
  200 C(1)        =   X(1)**2  +  X(6)**2
      CJAC(1,1)   =   TWO*X(1)
      CJAC(1,6)   =   TWO*X(6)
C
      C(2)        =   (X(2) - X(1))**2  +  (X(7) - X(6))**2
      CJAC(2,1)   = - TWO*(X(2) - X(1))
      CJAC(2,2)   =   TWO*(X(2) - X(1))
      CJAC(2,6)   = - TWO*(X(7) - X(6))
      CJAC(2,7)   =   TWO*(X(7) - X(6))
C
      C(3)        =   (X(3) - X(1))**2  +  X(6)**2
      CJAC(3,1)   = - TWO*(X(3) - X(1))
      CJAC(3,3)   =   TWO*(X(3) - X(1))
      CJAC(3,6)   =   TWO*X(6)
C
      C(4)        =   (X(1) - X(4))**2  +  (X(6) - X(8))**2
      CJAC(4,1)   =   TWO*(X(1) - X(4))
      CJAC(4,4)   = - TWO*(X(1) - X(4))
      CJAC(4,6)   =   TWO*(X(6) - X(8))
      CJAC(4,8)   = - TWO*(X(6) - X(8))
C
```

```
      C(5)        =   (X(1) - X(5))**2  +  (X(6) - X(9))**2
      CJAC(5,1)   =   TWO*(X(1) - X(5))
      CJAC(5,5)   = - TWO*(X(1) - X(5))
      CJAC(5,6)   =   TWO*(X(6) - X(9))
      CJAC(5,9)   = - TWO*(X(6) - X(9))
C
      C(6)        =   X(2)**2  +  X(7)**2
      CJAC(6,2)   =   TWO*X(2)
      CJAC(6,7)   =   TWO*X(7)
C
      C(7)        =   (X(3) - X(2))**2  +  X(7)**2
      CJAC(7,2)   = - TWO*(X(3) - X(2))
      CJAC(7,3)   =   TWO*(X(3) - X(2))
      CJAC(7,7)   =   TWO*X(7)
C
      C(8)        =   (X(4) - X(2))**2  +  (X(8) - X(7))**2
      CJAC(8,2)   = - TWO*(X(4) - X(2))
      CJAC(8,4)   =   TWO*(X(4) - X(2))
      CJAC(8,7)   = - TWO*(X(8) - X(7))
      CJAC(8,8)   =   TWO*(X(8) - X(7))
C
      C(9)        =   (X(2) - X(5))**2  +  (X(7) - X(9))**2
      CJAC(9,2)   =   TWO*(X(2) - X(5))
      CJAC(9,5)   = - TWO*(X(2) - X(5))
      CJAC(9,7)   =   TWO*(X(7) - X(9))
      CJAC(9,9)   = - TWO*(X(7) - X(9))
C
      C(10)       =   X(3)**2
      CJAC(10,3)  =   TWO*X(3)
C
      C(11)       =   (X(4) - X(3))**2  +  X(8)**2
      CJAC(11,3)  = - TWO*(X(4) - X(3))
      CJAC(11,4)  =   TWO*(X(4) - X(3))
      CJAC(11,8)  =   TWO*X(8)
C
      C(12)       =   (X(5) - X(3))**2  +  X(9)**2
      CJAC(12,3)  = - TWO*(X(5) - X(3))
      CJAC(12,5)  =   TWO*(X(5) - X(3))
      CJAC(12,9)  =   TWO*X(9)
C
      C(13)       =   X(4)**2  +  X(8)**2
      CJAC(13,4)  =   TWO*X(4)
      CJAC(13,8)  =   TWO*X(8)
C
      C(14)       =   (X(4) - X(5))**2  +  (X(9) - X(8))**2
      CJAC(14,4)  =   TWO*(X(4) - X(5))
      CJAC(14,5)  = - TWO*(X(4) - X(5))
      CJAC(14,8)  = - TWO*(X(9) - X(8))
      CJAC(14,9)  =   TWO*(X(9) - X(8))
C
      C(15)       =   X(5)**2  +  X(9)**2
      CJAC(15,5)  =   TWO*X(5)
      CJAC(15,9)  =   TWO*X(9)
      RETURN
C
C     END OF CONFUN
      END
```

```
ROWS OF  A.

  1       -1.00     1.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00

  2        0.00    -1.00     1.00     0.00     0.00     0.00     0.00     0.00     0.00

  3        0.00     0.00     1.00    -1.00     0.00     0.00     0.00     0.00     0.00

  4        0.00     0.00     0.00     1.00    -1.00     0.00     0.00     0.00     0.00

LOWER BOUNDS.
  0.00D-01 -1.00D 10 -1.00D 10 -1.00D 10  0.00D-01  0.00D-01  0.00D-01
 -1.00D 10 -1.00D 10  0.00D-01  0.00D-01  0.00D-01  0.00D-01 -1.00D 10
 -1.00D 10 -1.00D 10 -1.00D 10 -1.00D 10 -1.00D 10 -1.00D 10 -1.00D 10
 -1.00D 10 -1.00D 10 -1.00D 10 -1.00D 10 -1.00D 10 -1.00D 10 -1.00D 10

UPPER BOUNDS.
  1.00D 10  1.00D 10  1.00D 10  1.00D 10  1.00D 10  1.00D 10  1.00D 10
  0.00D-01  0.00D-01  1.00D 10  1.00D 10  1.00D 10  1.00D 10  1.00D 00
  1.00D 00  1.00D 00  1.00D 00  1.00D 00  1.00D 00  1.00D 00  1.00D 00
  1.00D 00  1.00D 00  1.00D 00  1.00D 00  1.00D 00  1.00D 00  1.00D 00

INITIAL  X.
      0.33      0.67      1.10      0.67      0.33      0.33      0.67
     -0.33     -0.67

WORKSPACE PROVIDED IS     IW(    50),  W(  1000).
TO SOLVE PROBLEM WE NEED  IW(    18),  W(   800).

 ITN ITQP     STEP NUMF   OBJECTIVE BND  LC  NC NCOLZ NORM GFREE  NORM ZTG   COND H    COND T    NORM C      RHO CONV U
   0   --  0.0D-01    1 -1.4400D 00  --  --  --    --   2.05D 00        --       --        --  9.36D-01        --   -- -
   1    8  4.0D-01    3 -1.4723D 00   0   0   5     4   2.08D 00  9.77D-02  1.6D 00   3.3D 00  6.98D-01  0.0D-01 FFFF 1
   2    8  1.0D 00    4 -1.3423D 00   0   0   4     5   2.00D 00  1.67D-01  2.2D 00   1.5D 00  6.91D-02  0.0D-01 FFTF 1
   3    1  1.0D 00    5 -1.3397D 00   0   0   4     5   2.05D 00  1.73D-01  6.9D 00   1.4D 00  4.57D-02  0.0D-01 FFTF 1
   4    3  1.4D 00    7 -1.3547D 00   0   0   6     3   2.09D 00  1.28D-01  6.9D 00   3.0D 00  1.11D-01  0.0D-01 FFTF 0
   5    1  1.0D 00    8 -1.3560D 00   0   0   6     3   2.06D 00  3.21D-02  5.5D 00   3.0D 00  1.69D-02  0.0D-01 FFTF 1
   6    1  1.0D 00    9 -1.3498D 00   0   0   6     3   2.05D 00  1.08D-02  5.6D 00   3.0D 00  2.37D-04  0.0D-01 FFTF 1
   7    1  1.0D 00   10 -1.3499D 00   0   0   6     3   2.05D 00  7.69D-03  7.5D 00   3.0D 00  1.29D-04  0.0D-01 FFTF 1
   8    1  1.0D 00   11 -1.3500D 00   0   0   6     3   2.05D 00  6.06D-03  1.5D 01   3.0D 00  2.43D-04  0.0D-01 FFTF 1
   9    1  1.0D 00   12 -1.3500D 00   0   0   6     3   2.05D 00  2.37D-03  2.8D 01   3.0D 00  1.05D-04  0.0D-01 FFTF 1
  10    1  1.0D 00   13 -1.3500D 00   0   0   6     3   2.05D 00  4.15D-04  3.0D 01   3.0D 00  4.85D-06  0.0D-01 FFTF 1
  11    1  1.0D 00   14 -1.3500D 00   0   0   6     3   2.05D 00  4.57D-05  2.4D 01   3.0D 00  8.11D-08  0.0D-01 FFTF 1
  12    1  1.0D 00   15 -1.3500D 00   0   0   6     3   2.05D 00  7.40D-06  2.3D 01   3.0D 00  3.17D-09  0.0D-01 FTTF 1
  13    1  1.0D 00   16 -1.3500D 00   0   0   6     3   2.05D 00  7.43D-07  2.3D 01   3.0D 00  3.40D-11  0.0D-01 FTTF 1
  14    1  1.0D 00   17 -1.3500D 00   0   0   6     3   2.05D 00  2.89D-08  1.8D 01   3.0D 00  6.07D-13  0.0D-01 TTTF 1
  15    1  1.0D 00   18 -1.3500D 00   0   0   6     3   2.05D 00  1.81D-09  1.4D 01   3.0D 00  9.44D-15  0.0D-01 TTTT 1

EXIT NP PHASE.   INFORM =  0   MAJITS =  15  NFEVAL =  18  NCEVAL =  18

VARIABLE          STATE      VALUE       LOWER BOUND     UPPER BOUND   LAGR MULTIPLIER       RESIDUAL

VARBL  1            FR   0.6094665D-01   0.0000000          NONE          0.0000000         0.6095D-01
VARBL  2            FR   0.5976493            NONE          NONE          0.0000000         0.1000D 11
VARBL  3            FR    1.000000            NONE          NONE          0.0000000         0.1000D 11
VARBL  4            FR   0.5976493            NONE          NONE          0.0000000         0.1000D 11
VARBL  5            FR   0.6094665D-01   0.0000000          NONE          0.0000000         0.6095D-01
```

VARBL 6	FR	0.3437715	0.0000000	NONE	0.0000000	0.3438
VARBL 7	FR	0.5000000	0.0000000	NONE	0.0000000	0.5000
VARBL 8	FR	-0.5000000	NONE	0.0000000	0.0000000	0.5000
VARBL 9	FR	-0.3437715	NONE	0.0000000	0.0000000	0.3438

LINEAR CONSTR	STATE	VALUE	LOWER BOUND	UPPER BOUND	LAGR MULTIPLIER	RESIDUAL
LNCON 1	FR	0.5367027	0.0000000	NONE	0.0000000	0.5367
LNCON 2	FR	0.4023507	0.0000000	NONE	0.0000000	0.4024
LNCON 3	FR	0.4023507	0.0000000	NONE	0.0000000	0.4024
LNCON 4	FR	0.5367026	0.0000000	NONE	0.0000000	0.5367

NONLNR CONSTR	STATE	VALUE	LOWER BOUND	UPPER BOUND	LAGR MULTIPLIER	RESIDUAL
NLCON 1	FR	0.1218933	NONE	1.000000	0.0000000	0.8781
NLCON 2	FR	0.3124571	NONE	1.000000	0.0000000	0.6875
NLCON 3	UL	1.000000	NONE	1.000000	-0.8318406D-01	-0.4219D-14
NLCON 4	UL	1.000000	NONE	1.000000	-0.3202625	-0.3997D-14
NLCON 5	FR	0.4727152	NONE	1.000000	0.0000000	0.5273
NLCON 6	FR	0.6071847	NONE	1.000000	0.0000000	0.3928
NLCON 7	FR	0.4118861	NONE	1.000000	0.0000000	0.5881
NLCON 8	UL	1.000000	NONE	1.000000	-0.1992983	-0.3997D-14
NLCON 9	UL	1.000000	NONE	1.000000	-0.3202625	-0.4441D-14
NLCON 10	UL	1.000000	NONE	1.000000	-0.3437715	0.0000
NLCON 11	FR	0.4118861	NONE	1.000000	0.0000000	0.5881
NLCON 12	UL	1.000000	NONE	1.000000	-0.8318406D-01	-0.4441D-14
NLCON 13	FR	0.6071847	NONE	1.000000	0.0000000	0.3928
NLCON 14	FR	0.3124571	NONE	1.000000	0.0000000	0.6875
NLCON 15	FR	0.1218933	NONE	1.000000	0.0000000	0.8781

EXIT NPSOL - OPTIMAL SOLUTION FOUND.

FINAL NONLINEAR OBJECTIVE VALUE = -1.349963

A RUN OF THE SAME EXAMPLE WITH A WARM START....

```
INITIAL  X.
     0.11      0.65      1.05      0.65      0.11      0.39      0.55
    -0.45     -0.29

WORKSPACE PROVIDED IS     IW(    50),  W(  1000).
TO SOLVE PROBLEM WE NEED  IW(    18),  W(   800).
```

ITN	ITQP	STEP	NUMF	OBJECTIVE	BND	LC	NC	NCOLZ	NORM GFREE	NORM ZTG	COND H	COND T	NORM C	RHO	CONV	U
0	--	0.0D-01	1	-1.3843D 00	--	--	--	--	2.09D 00	--	--	--	1.14D-01	--	--	-
1	1	2.2D 00	3	-1.3148D 00	0	0	6	3	2.03D 00	5.39D-02	5.3D 02	2.0D 00	1.21D-01	0.0D-01	FFTF	1
2	1	1.0D 00	4	-1.3517D 00	0	0	6	3	2.06D 00	1.35D-02	9.9D 02	1.9D 00	5.07D-03	0.0D-01	FFTF	1
3	1	1.0D 00	5	-1.3499D 00	0	0	6	3	2.05D 00	1.08D-02	7.9D 01	3.0D 00	4.37D-05	0.0D-01	FFTF	1
4	1	1.0D 00	6	-1.3500D 00	0	0	6	3	2.05D 00	4.39D-03	7.8D 01	3.0D 00	1.52D-05	0.0D-01	FFTF	1
5	1	1.0D 00	7	-1.3500D 00	0	0	6	3	2.05D 00	1.14D-03	4.1D 01	3.0D 00	2.12D-05	0.0D-01	FFTF	1
6	1	1.0D 00	8	-1.3500D 00	0	0	6	3	2.05D 00	3.13D-04	1.9D 02	3.0D 00	7.22D-06	0.0D-01	FFTF	1
7	1	1.0D 00	9	-1.3500D 00	0	0	6	3	2.05D 00	4.94D-05	5.2D 02	3.0D 00	3.19D-07	0.0D-01	FFTF	1
8	1	1.0D 00	10	-1.3500D 00	0	0	6	3	2.05D 00	2.80D-06	7.4D 02	3.0D 00	1.25D-09	0.0D-01	FTTF	1
9	1	1.0D 00	11	-1.3500D 00	0	0	6	3	2.05D 00	5.41D-08	6.9D 02	3.0D 00	9.12D-12	0.0D-01	TTTF	1
10	1	1.0D 00	12	-1.3500D 00	0	0	6	3	2.05D 00	9.16D-10	5.4D 02	3.0D 00	1.83D-14	0.0D-01	TTTT	1

EXIT NP PHASE. INFORM = 0 MAJITS = 10 NFEVAL = 12 NCEVAL = 12

EXIT NPSOL - OPTIMAL SOLUTION FOUND.

FINAL NONLINEAR OBJECTIVE VALUE = -1.349963

REFERENCES

Biggs, M. C. (1972). "Constrained minimization using recursive equality quadratic programming", in *Numerical Methods for Non-Linear Optimization* (F. A. Lootsma, ed.), pp. 411–428, Academic Press, London and New York.

Fletcher, R. (1981). *Practical Methods of Optimization, Volume 2, Constrained Optimization*, John Wiley and Sons, New York and Toronto.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1982). Procedures for optimization problems with a mixture of bounds and general linear constraints, Report SOL 82-6, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1983a). User's guide for SOL/QPSOL, Report SOL 83-7, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1983b). The design and implementation of a quadratic programming algorithm, to appear, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1983c). User's guide for SOL/LCSOL, to appear, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press, London and New York.

Han, S.-P. (1976). Superlinearly convergent variable metric algorithms for general nonlinear programming problems, *Math. Prog.* **11**, pp. 263–282.

Hock, W. and Schittkowski, K. (1981). *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer-Verlag, Berlin and New York.

Murtagh, B. A. and Saunders, M. A. (1980). MINOS/AUGMENTED User's Manual, Report SOL 80-14, Department of Operations Research, Stanford University, California.

Murtagh, B. A. and Saunders, M. A. (1982). A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints, *Math. Prog. Study* **16**, pp. 84–118.

Powell, M. J. D. (1977). A fast algorithm for nonlinearly constrained optimization calculations, Report DAMTP 77/NA 2, University of Cambridge, England.

Powell, M. J. D. (1982). State-of-the-art tutorial on variable metric methods for constrained optimization, Report DAMTP 1982/NA5, University of Cambridge, England.

Ryder, B. G. (1974). The PFORT verifier, *Software-Practice and Experience* **4**, pp. 359–377.

Schittkowski, K. (1981). The nonlinear programming method of Wilson, Han and Powell with an augmented Lagrangian type line search function. Part 2: An efficient implementation with linear least squares subproblems, *Numer. Math.* **38**, pp. 115–127.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 83-12	2. GOVT ACCESSION NO. A132951	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) User's Guide for SOL/NPSOL: A Fortran Package for Nonlinear Programming		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Philip E. Gill, Walter Murray, Michael A. Saunders and Margaret H. Wright		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0267 DAAG29-81-K-0156
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		12. REPORT DATE July 1983
		13. NUMBER OF PAGES pp.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

THE VIEW, OPINIONS, AND/OR FINDINGS CONTAINED IN THIS REPORT ARE THOSE OF THE AUTHOR(S) AND SHOULD NOT BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION, POLICY, OR DECISION, UNLESS SO DESIGNATED BY OTHER DOCUMENTATION.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

optimization
non-linear programming
sequential quadratic programming method
mathematical software

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 83-12: User's Guide for SOL/NPSOL: A Fortran Package for Nonlinear Programming, by Philip E. Gill, Walter Murray, Michael A. Saunders and Margaret H. Wright

This report forms the user's guide for version 1.1 of SOL/NPSOL, a set of Fortran subroutines designed to minimize an arbitrary smooth function subject to constraints, which may include simple bounds on the variables, linear constraints and smooth nonlinear constraints. (NPSOL may also be used for unconstrained, bound-constrained and linearly constrained optimization.) The user must provide subroutines that define the objective and constraint functions and their gradients. All matrices are treated as dense, and hence NPSOL is not intended for large sparse problems.

NPSOL uses a sequential quadratic programming (SQP) algorithm, in which the search direction; is the solution of a quadratic programming (QP) subproblem. The algorithm treats bounds, linear constraints and nonlinear constraints separately. The Hessian of each QP subproblem is a positive-definite quasi-Newton approximation to the Hessian of an augmented Lagrangian function. The steplength at each iteration is required to produce a sufficient decrease in an augmented Lagrangian merit function. Each QP subproblem is solved using a quadratic programming package with several features that improve the efficiency of an SQP algorithm.

END

FILMED

10-83

DTIC